Laurent NGUYEN

20 Décembre 2019

Compte – rendu sur le cours d'Application Web

A travers ce document, nous allons voir les différentes notions sur les applications web en java, notamment à travers les pages jsp, le JavaBean ou encore les servlets. Nous allons traiter chaque sujet avec quelques exemples très simples. Puis nous finirons avec un aperçu d'un projet en python afin d'avoir un cas concret d'utilisation d'API REST.

Partie 1 : Initialisation du projet et lancement

1- Configurations nécessaires

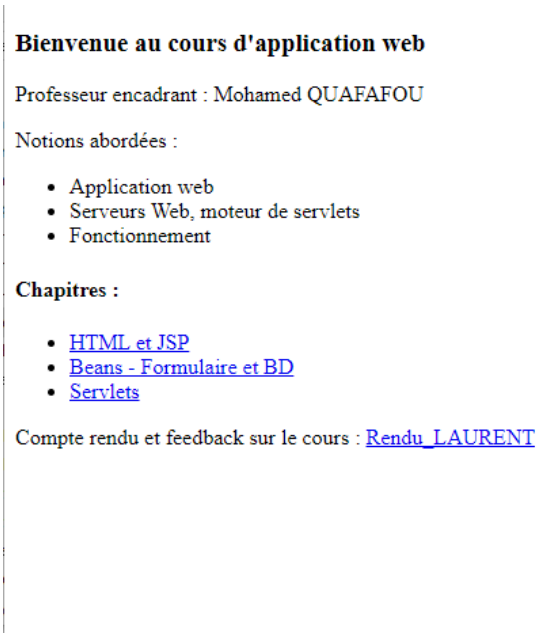
Afin de récupérer notre projet puis de l'exécuter, il est nécessaire d'avoir configuré notre environnement avec ces outils :

- Eclipse : *https://www.eclipse.org/*
- JDK 8+ : https://www.oracle.com/technetwork/java/javase/downloads/index.html
- Serveur Tomcat 7+ : https://tomcat.apache.org
- Le projet sur GitHub : https://github.com/NgLaurent/AppWeb

2- Démarrage du serveur tomcat sur Eclipse

Afin de pouvoir visualiser notre projet, il faut :

- Lancer Eclipse
- Importer notre projet dans votre workspace
- Ajouter un serveur Apache tomcat
    - Faire new > Server > apachetomcat > Aller chercher votre fichier apache-tomcat que vous avez téléchargé précédemment > Finish
- Lancer le serveur
    - Bouton Run server on localhost
- Aller sur la page http://localhost:8080/appli-web/index.html



Bienvenue au cours d'application web

Professeur encadrant : Mohamed QUAFAFOU

Notions abordées :

- Application web
- Serveurs Web, moteur de servlets
- Fonctionnement

Chapitres :

- HTML et JSP
- Beans - Formulaire et BD
- Servlets

Compte rendu et feedback sur le cours : Rendu_LAURENT

Partie 2 : JSP, Beans et Servlet

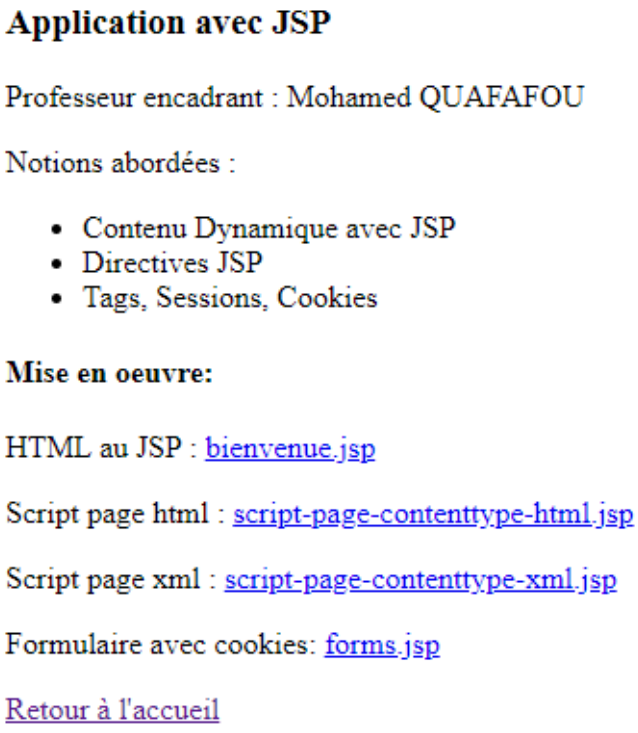
1- Les pages JSP

Les pages JSP sont des pages HTML mais avec du code java. Le code HTML reste le même mais il est possible d'inclure du code java à l'aide de balises <% %>. Nous pouvons aussi importer des classes java dans notre code à partir de la balise <@ page import= « java.util.* » > par exemple. Il existe aussi quelques directives intéressantes à connaitre :

- <%@ include file = « xxx.jsp » %> qui permet d'inclure le contenu de xxx.jsp directement dans notre page
- <jsp:forward page="display.jsp"> qui permet de rediriger vers une autre page (ici display)

Sur notre application vous pourrez visualiser les exemples suivants :

- Affichage simple d'une page jsp contenant la date et un tableau (généré avec du code java)
- Affichage d'une page en format html ou xml
- Gestion d'un formulaire avec cookies



Application avec JSP

Professeur encadrant : Mohamed QUAFAFOU

Notions abordées :

- Contenu Dynamique avec JSP
- Directives JSP
- Tags, Sessions, Cookies

Mise en oeuvre:

HTML au JSP : bienvenue.jsp

Script page html : script-page-contenttype-html.jsp

Script page xml : script-page-contenttype-xml.jsp

Formulaire avec cookies: forms.jsp

Retour à l'accueil

Page Jsp.html

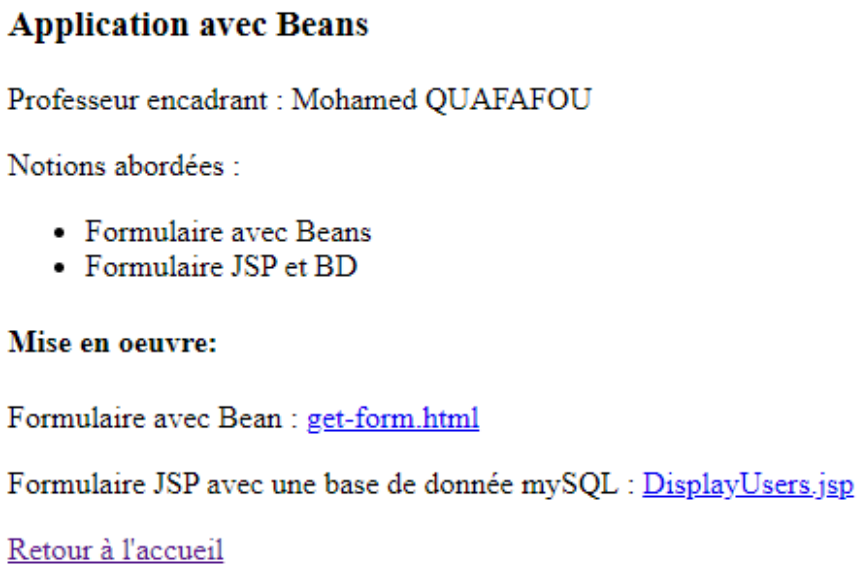
2- Les JavaBeans

Les Beans sont des POJO (Plain Old Java Object) qui nous permettent de créer des modèles d'objets que nous pourrons ensuite facilement utiliser. Par exemple, pour un formulaire de création de compte sur une application quelconque, il est intéressant de créer un bean comportant les réponses de l'utilisateur pour ensuite les réutiliser de façon efficace à partir des getters.
Pour définir et utiliser un bean :

- Nous créons notre POJO avec les champs nécessaires et les getteurs/setteurs
- Nous mappons notre objet à un formulaire par exemple
- Nous récupèrons notre bean sur une autre page puis nous l'utilisons simplement à partir de ses getteurs

Sur notre application vous pourrez visualiser les exemples suivants :

- Formulaire en utilisant un Bean
- Formulaire en utilisant un Bean et une base de données MySQL



Page Beans.html

3- Les Servlets

Les servlets nous permettent de créer des chemins (« /login » par exemple) pour notre application afin de gérer des requêtes HTTP de type GET, POST, DELETE, PUT. Les servlets sont très intéressants et utiles à utiliser car elles prennent en paramètre deux objets de type :

- HttpServletRequest qui permet de récupérer le contenu de la requête http, ou encore les paramètres dans l'url de type « ?username=Laurent&pw=mdp »
- HttpServletResponse qui permet de modifier le contenu de la réponse du servlet. On peut ainsi modifier le code http de la réponse et le contenu de ce que l'on veut renvoyer.

Tous ces traitements sont faits dans les méthodes doGet, DoPost etc..

Sur notre application vous pourrez visualiser les exemples suivants :

- Authentification via servlet
- Cycle de vie d'un servlet
- Formulaire à choix multiples
- Formulaire complet utilisant tous les notions vues précédemment

Application avec Servlets

Professeur encadrant : Mohamed QUAFAFOU

Notions abordées :

- Utilisation de servlets
- Formulaire avec Servlets

Mise en oeuvre:

Servlet simple affiche bienvenue et le nombre de fois qu'il a été appelée: /servlet

Affichage de la date courante (qui s'actualise tout seul) avec servlet : /date

Authentification via servlet formulaire sur /form : Formulaire d'Authentification

Creation et verification d'une session crée avec servlet : /session

Cycle de vie d'un servlet : /cycle

Formulaire à choix multiple : Choix de joueur de foot avec servlet

Formulaire complet avec jsp / beans / servlet : Formulaire complet

Retour à l'accueil

Page Servlet.html

Partie 3 : Projet Python

1- Exemple avec Twitch

Afin de mieux comprendre les appels http automatiques pour modéliser et implémenter le projet scolar que nous expliquerons par la suite, nous avons fait le tutoriel suivant :

Projet Scrapy Selenium Twitch : http://mroseman.com/scraping-dynamic-pages/

Ce projet a été développé en Python et utilise deux frameworks :

- Scrapy, permettant de scruter une page HTML et y extraire des informations
- Selenium, permettant d'automatiser la récupération de page HTML

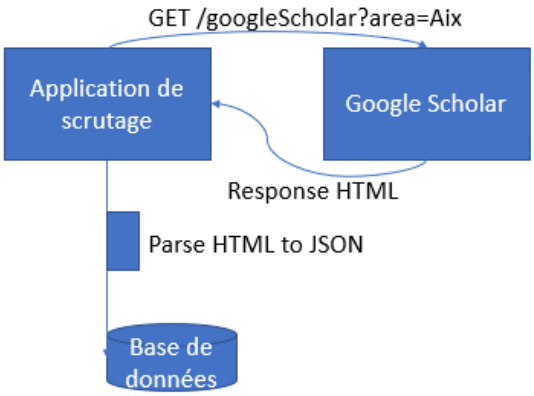
Dans ce tutoriel, l'application fait un appel simple sur une page de twitch (plateforme de streaming) et scrute la page afin de récupérer des informations qui seront ensuite stockées dans un fichier json.
Nous pouvons alors comprendre comment certaines pages d'e-commerce modernes fonctionnent pour fixer des prix dynamiques. En effet, en implémentant une application de scrutage automatisée, un site d'e-commerce peut alors faire le scrutage automatique d'un produit de son catalogue sur les sites d'e-commerce concurrent afin de récupérer les prix du marché et ainsi fixer un prix dynamique plus intéressant. Attention, ce genre de pratique peut très vite être détecté : il est donc important d'automatiser ces scrutages sur des intervalles bien espacés pour éviter que le site d'e-commerce concurrent détecte qu'il s'agit d'un appel M2M (Machine to Machine).

2- Projet Google Scholar

A partir de l'exemple précèdent, nous pouvons ainsi modéliser un service que nous pourrons exposer dans notre école. Ce projet se découpe en deux parties :

1) <u>La récupération des données</u>

Dans cette partie, nous pouvons automatiser des appels HTML (avec des intervalles bien élargis) vers google scholar afin de remplir notre base de données.
Langage : Python
Automatisation des requêtes : Selenium
Scrutage des pages : Scrapy
Base de données noSQL : MongDB

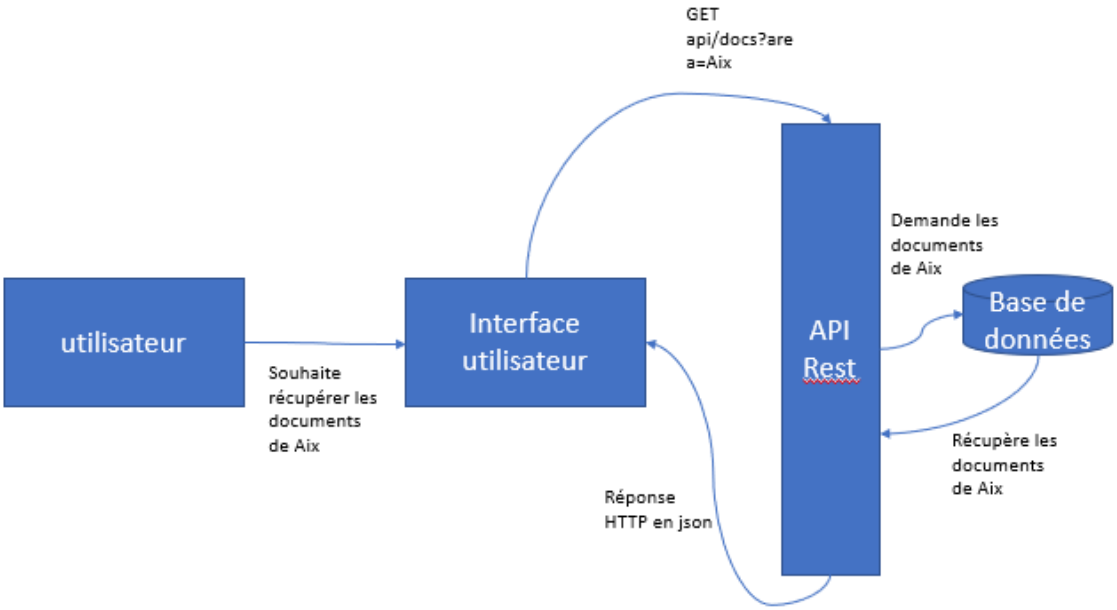


Remplissage des documents de Aix sur google scholar dans notre base de données

1) L'exposition des données

Après avoir recuperé l'ensemble des données de Google Scholar, nous pouvons enfin exposer une API RESTFul pour permettre à des applications ou utilisateurs de consommer ces données à partir d'une simple requête http.

L'implementation pourra se faire à partir d'un serveur nodejs avec le framework Express, et utiliser Mongoose pour faciliter la connection à la base de donnée MongoDB. Nous pouvons aussi penser à une interface utilisateur que nous pourrons developper en Angular afin d'avoir une application web robuste en MEAN (MongoDB Express Angular NodeJs) stack.



Exposition de données à partir d'une API REST